RESOLUCIÓN DE PROBLEMAS Y ALGORITMOS

CLASE 6

Repetición incondicional
(sentencia FOR)

Luciano H. Tamargo
http://cs.uns.edu.ar/~lt
Depto. de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur, Bahía Blanca
2016

UNIDADES DE ALMACENAMIENTO

Unidad	Equivale a
1 Bit	
1 Byte	8 bits
1 Kilobyte (KB)	1000 bytes
1 Megabyte (MB)	1000 KB = 1.000.000 bytes
1 Gigabyte (GB)	1000 MB = 1.000.000.000 bytes
1 Terabyte (TB)	1000 GB = 1.000.000.000.000 bytes
1 Petabyte (PB)	1000 TB = 1.000.000.000.000.000 bytes
1 Exabyte (EB)	1000 PB = 1.000.000.000.000.000.000 bytes
1 Zettabyte (ZB)	1000 EB = 1.000.000.000.000.000.000.000 bytes
1 Yottabyte (YB)	1000 ZB = 1.000.000.000.000.000.000.000.000 bytes

Más detalles en https://en.wikipedia.org/wiki/Binary_prefix

EJEMPLOS

Unidad
1 Bit
1 Byte
1 Kilobyte (KB)
1 Megabyte (MB)
1 Gigabyte (GB)
1 Terabyte (TB)
1 Petabyte (PB)
1 Exabyte (EB)
1 Zettabyte (ZB)
1 Yottabyte (YB)

- Un código fuente en Pascal (prueba.pas) de un programa sencillo ocupa 134 bytes.
- El ejecutable de ese programa ocupa 69 KB.
- El archivo pdf de la clase pasada ocupa 1,3 MB.
- La memoria RAM de mi notebook es de 4 GB.
- En 2015 un disco rígido típico era de 1 TB.
- El contenido de la web en 1995 se estimaba en 8 PB.
- El contenido de la web en 2009 se estimaba en 500EB.
- En un gramo de ADN contiene unos 455 EB de información.
- El contenido de la web en 2013 se estimaba en 4 ZB (¿unos 8 gramos de ADN?).

¿Más curiosidad? vea:
https://en.wikipedia.org/wiki/Yottabyte

CONCEPTOS Y VOCABULARIO

- En un lenguaje de programación, las sentencias repetitivas permiten repetir un cierto número de veces un grupo (bloque) de sentencias.
- A la **repetición** también se la llama **iteración**.
- Al repetir (o iterar) un grupo de sentencias, este grupo vuelve a comenzar una y otra vez.
- Es por esto que en la jerga informática, para hablar de una iteración también se usan las palabra **ciclo**, **lazo** o **bucle**. *En inglés: **loop**.*

Repetir 10 veces:
Llenar botella
Pasar a bidón

El grupo:
"llenar botella
pasar a bidón"
se ejecutará en secuencia 10 veces.

CONCEPTOS Y VOCABULARIO

- En **Pascal existen tres sentencias diferentes** para realizar repetición.
- **Por ejemplo**: en la siguiente iteración, la secuencia de sentencias (sentencia compuesta) dentro del ciclo **FOR** se ejecutará 10 veces.

```
FOR V:= 1 TO 10 DO
  begin
    … secuencia de sentencias…
  end;
```

Ciclo FOR o bucle FOR.
En inglés: "FOR-DO Loop".

Repetir 10 veces:
Llenar botella
Pasar a bidón

El grupo:
"llenar botella
pasar a bidón"
se ejecutará en secuencia 10 veces.

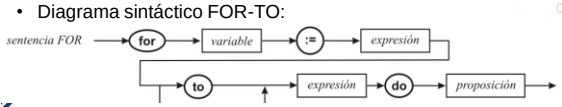
CONCEPTO: REPETICIÓN INCONDICIONAL EN PASCAL

- Sentencia **FOR** de Pascal:

La sentencia se ejecuta un **número fijo de veces** y luego continua en:

```
FOR V:= valor_inicial TO valor_final DO
  1 sentencia simple
  o compuesta ;
Otra sentencia siguiente;
```

- Diagrama sintáctico FOR-TO:



CONCEPTO: REPETICIÓN INCONDICIONAL EN PASCAL

- Las sentencias de un ciclo **FOR-TO** se ejecutan CERO o más veces dependiendo de *valor_inicial* y *valor_final*.

La sentencia se ejecuta un **número fijo de veces** y luego continua en:

```
FOR V:= valor_inicial TO valor_final DO
  1 sentencia simple
  o compuesta ;
Otra sentencia siguiente;
```

- V debe ser una variable de tipo ordinal (que se suele llamar **variable de control**).
- *valor_inicial* y *valor_final* son expresiones cuyo valor resultante debe pertenecer al mismo tipo que la variable V.
- Al comenzar, a V se le asigna *valor_inicial*.
- Luego, V es **incrementada automáticamente de a uno** en cada repetición (hasta llegar a *valor_final*).

CONCEPTOS: TIPOS ORDINALES EN PASCAL

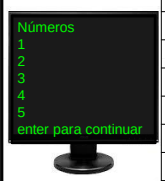
Tipo de Dato: define el conjunto de valores posibles que puede tomar una variable, las operaciones que pueden aplicarse, y cual es la representación interna.

- De los 4 tipos simples predefinidos que hemos visto, **INTEGER, CHAR** y **BOOLEAN** son **tipos ordinales** (REAL no es ordinal).
- Los **tipos ordinales** poseen estas características:
  - Tienen un orden. Tienen un primer y último elemento.
  - Para cada elemento está definido el siguiente (a excepción del último) y el anterior (a excepción del primero).
  - Tienen definidas las operaciones predefinidas:
    - **succ()** retorna el siguiente (excepto del último)
    - **pred()** retorna el anterior (excepto del primero)
    - **ord()** retorna el número de orden

EJEMPLO

- Escriba un programa para mostrar por pantalla todos los números enteros entre 1 y 5.

```
PROGRAM ListaNumeros;
{Muestra todos los números enteros
entre  1 y 5}
VAR
  num:INTEGER;
BEGIN
writeln('Números');
FOR  num:= 1  TO 5 DO
  writeln(num);
writeln('enter para continuar');
readln; // mantiene abierta consola
END.
```



num
?
1
2
3
4
5

REPETICIÓN INCONDICIONAL

```
FOR V:= valor_inicial TO valor_final DO
  sentencia simple o compuesta
```

- *valor_inicial* y *valor_final* son expresiones cuyo valor debe pertenecer al mismo tipo que la variable de control **V.**
- La *sentencia* (que puede ser compuesta), se repetirá un número fijo de veces: *valor_final* – *valor_inicial* + 1.

```
FOR V:= 1 TO (2*2)+1 DO
  writeln(V);
```

repite 5 veces

- Si *valor_final* **es menor estricto** a *valor_inicial* entonces se repetirá 0 veces.

```
FOR V:= 5 TO 1 DO
  writeln(V);
```

repite 0 veces

REPETICIÓN INCONDICIONAL

- Al comenzar, a V se le asigna el valor inicial y luego, V se incrementada **automáticamente de a uno** hasta llegar al valor final.

```
FOR V:= 1 TO 100 DO
  <sentencia>
```

Aquí <sentencia> se repite 100 veces: 100-1+1

```
FOR V:= 100 TO 199 DO
  <sentencia>
```

Aquí <sentencia> se repite 100 veces: 199-100+1

```
FOR V:= -10 TO -1 DO
  <sentencia>
```

Aquí <sentencia> se repite 10 veces: -1-(-10)+1

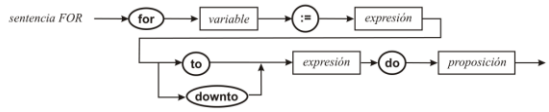
```
FOR V:= 1 TO -2 DO
  <sentencia>
```

Aquí <sentencia> se repite 0 veces

SENTENCIA FOR-DOWNTO

```
FOR V:= exp1 DOWNTO exp2 DO
  <sentencia>
```

- *exp1* y *exp2* pueden ser valores, variables, o expresiones del mismo tipo ordinal que la variable V *(no puede ser tipo real).*




```
FOR numero:= 10 DOWNTO 0 DO
  write(numero);
```

```
FOR letra:= 'Z' DOWNTO 'A' DO
  write(letra);
```

SENTENCIA FOR-DOWNTO

```
FOR V:= exp1 DOWNTO exp2 DO
  <sentencia>
```

- El resultado de *exp1* y *exp2* deben poder calcularse justo antes de la ejecución del bucle FOR-DOWNTO. La variable V toma el valor inicial de evaluar *exp1* y luego, en cada iteración, V se **decrementa automáticamente de a uno** hasta llegar al valor de *exp2*.
- Si *val1* es el valor de *exp1* y *val2* el de *exp2*, entonces la *sentencia*, que puede ser compuesta, se repetirá (*val1* – *val2* + 1) veces.
- Si *val2* **es mayor estricto** a *val1* entonces se repetirá 0 veces.

```
FOR num:= 5 DOWNTO 1 DO
  write(num);
```
Repite 5 veces

```
FOR num:= 1 DOWNTO 5 DO
  write(num);
```
Repite 0 veces

- **Importante**: La sentencia FOR incrementa o decrementa automáticamente de a un valor.

EJEMPLO

- Escriba un programa para mostrar por pantalla todos los números enteros entre dos topes ingresados.

```
PROGRAM ListaNumeros;
{Muestra todos los números enteros
desde tope inferior a tope
superior}
VAR
topeinf, topesup, num:INTEGER;
BEGIN
 writeln('Ingrese los topes:');
 readln(topeinf, topesup);
 writeln('Números');
 FOR num:=topeinf TO topesup DO
   write(num,' , ');
END.
```

topeinf	toesup	num
?	?	?
3	8	3
		4
		5
		6
		7
		8

```
Ingrese los
topes:
3 8
Números
3,4,5,6,7,8,
```


EJEMPLO

- Escriba un programa para mostrar por pantalla todos los números enteros entre dos topes ingresados.

```
PROGRAM ListaNumeros;
{Muestra todos los números enteros desde
tope inferior a tope superior}
VAR
topeinf, topesup, num:INTEGER;
BEGIN
 writeln('Ingrese los topes:');
 readln(topeinf, topesup);
 writeln('Números');
 FOR num:=topeinf TO topesup-1 DO
   write(num,' , ');
 Write(topesup) {Escribe por separado el
 último entero para evitar la coma final}
END.
```

```
Ingrese los
topes:
3 8
Números
3,4,5,6,7,8
```


EJEMPLO

- Escriba un programa para mostrar por pantalla todos los números entre dos topes ingresados en cualquier orden.

```
PROGRAM ListaNumeros; {muestra los números entre dos topes}
VAR
topeinf, topesup, num, aux: INTEGER;
BEGIN
writeln('Ingrese topes: ');
readln(topeinf, topesup);
IF topesup < topeinf THEN {si los topes están invertidos}
  begin {intercambio los valores de los topes}
    aux := topeinf;
    topeinf := topesup;
    topesup := aux;
  end;
writeln('Números');
FOR num:= topeinf TO topesup -1 DO
  write(num,' , ');
write(topesup)
END.
```

Casos de prueba:
3 6
6 3
6 6

```
Ingrese topes:
6 3
Números
3,4,5,6
```

SENTENCIA FOR-TO

```
FOR V:= exp1 DOWNTO exp2 DO
  <sentencia>
```

- Tanto *exp1* como *exp2* pueden ser valores, variables, o expresiones siempre que sean del mismo tipo ordinal que V (**no puede ser tipo real**).
- Al comenzar, a V se le asigna el valor de evaluar *exp1* y luego, V se incrementada **automáticamente de a uno** hasta llegar al valor de *exp2*.

```
N:=2;
FOR V := 1+1 TO N DO
  writeln(V);
FOR V := N+N TO sqr(N+N) DO
  writeln(V);
FOR V := N div 2 TO (N+10) - N + 2 DO
  writeln(V);
```


FOR-TO CON TIPO CHAR

```
PROGRAM letras_y_codigos;
{muestra en pantalla los códigos
ASCII de algunas letras en columnas}
CONST
ultima='J';
columnas=3;
VAR
letra: char;
contador: integer;
BEGIN
contador:=1;
FOR letra:= 'A' TO ultima DO
  BEGIN {comienzo del ciclo for}
    write(letra,'=', ord(letra),' ');
  {baja de renglón cada "columnas" veces}
    if contador mod columnas = 0 then
      writeln;
    contador:=contador + 1;
  END {fin del ciclo FOR}
END.
```

```
A=65  B=66  C=67
D=68  E=69  F=70
G=71  H=72  I=73
J=74
```

Comienza con el valor 'A' y luego incrementa automáticamente de a uno pasando por todos los valores del código ASCII hasta llegar al valor final ('J')

FOR-DOWNTO. EJEMPLO.

```
PROGRAM CuentaRegresiva;
{Imprime en pantalla los dígitos
de 9 a 0 en ese orden y las
letras de G a A en ese orden}
VAR
dig:integer;
letra:char;
BEGIN
writeln('Cuenta regresiva');
FOR dig:=9 DOWNTO 0 DO
  write(dig,' ');
writeln;
writeln('----------------');
FOR letra:='G' DOWNTO 'A' DO
  write(letra,' ');
writeln;
writeln('----------------');
END.
```

```
Cuenta regresiva
9 8 7 6 5 4 3 2 1 0
----------------
G F E D C B A
----------------
```

dig	letra
?	?

PROBLEMA PROPUESTO

- Escriba un programa que calcule el promedio de una cantidad conocida (e ingresada por el usuario) de números reales.

```
Ingrese la cantidad de valores:
4
Ingrese un valor: 8.2
Ingrese un valor: 0.2
Ingrese un valor: -3.0
Ingrese un valor: 5.2
El promedio es: 2.65
```


PROBLEMA PROPUESTO

- Escriba un programa que calcule el promedio de una cantidad conocida (e ingresada por el usuario) de números reales.
- **Solución**: El promedio consiste de sumar todo los números reales ingresados y dividir por la cantidad ingresada.
- La dificultad que presenta no conocer hasta el momento de la ejecución cuantos valores hay que sumar, se resuelve calculando la suma a medida que los valores son ingresados (sin guardar los valores individuales) .

PROBLEMA PROPUESTO

Igoritmo "promedio":
  Leer cantidad
  Suma ← 0
  Repetir cantidad veces
    leer valor
    suma ← valor + suma
  Mostrar suma/cantidad

La flecha "←" es el símbolo de asignación usualmente usado en algoritmos.

El dato "suma" (inicialmente en cero) va acumulando el resultado de sumar los valores leídos.

UN PROGRAMA POSIBLE PARA PROMEDIO

```
PROGRAM Promedio;
{Calcula el promedio de una cant. fija de
valores reales ingresados}
VAR
cantidad, control: integer;
suma, valor, prom: real;
BEGIN
writeln('Cantidad de valores:');
readln(cantidad);
suma:=0;//valor inicial
FOR control:= 1 TO cantidad DO
 begin
  writeln('Ingrese un valor: ');
  readln(valor);
  //cada valor leído borra el anterior
  suma:=suma+valor;//pero la suma se acum
 end;
prom:= suma/cantidad;
writeln('El promedio es:',prom:0:2);
END.
```

TRAZA

cantidad	suma	valor	control	prom
?	?	?	?	?
3	0	?	?	?
	8	8	1	?
	15	7	2	?
	24	9	3	8

PROBLEMA PROPUESTO

- Escriba un programa para calcular un número natural elevado a una potencia (también natural).
- **Ejemplos**:

  $2^3=8$ $3^2=9$ $1^6=1$ $2^{10}=1024$

  $2^3=2*2*2=8$ $3^0=1$ $3^2=3*3=9$ $1^6=1*1*1*1*1*1=1$
- **Solución**: multiplicar "base" "exponente" veces

Algoritmo "potencia":
  Leer base y exponente
  Potencia ← 1
  Repetir exponente veces
    potencia ← potencia * base
  Mostrar potencia

UN POSIBLE PROGRAMA PARA "POTENCIA"

```
PROGRAM CalculoPotencia;
{Calcula un número natural elevado a
una potencia (también natural)}
VAR
base, exponente,
potencia, aux: integer;
BEGIN
writeln('Ingrese base y exp : ');
readln(base, exponente);
potencia := 1;
FOR aux := 1 TO exponente DO
  potencia := potencia * base;
write(base, ' a la ', exponente);
writeln(' es ',  potencia);
END.
```

```
Ingrese base y exp:
2 7
2 a la 7 es 128
```

Base	exponente	aux	potencia
?	?	?	?
2	7	1	2
		2	4
		3	8
		4	16
		5	32
		6	64
		7	128

PROBLEMA PROPUESTO

- Factorial de un número N se denota con N!
  y se define: N! = 1 * 2 * 3 * 4 * ..... * N
  0! = 1

Ejemplos:
1! = 1
2! = 2 3! = 6 4! = 24 5! = 120
6! = 720
7! = 5.040
8! = 40.320
9! = 362.880
10! = 3.628.800

Problema: Escriba un programa para calcular el Factorial de un número N ingresado por el usuario.

ALGORITMO PROPUESTO

Problema: Escriba un programa para calcular el Factorial de un número N ingresado por el usuario.

- Como N! = 1 * 2 * 3 * 4 * ..... * N
- Por lo tanto tengo que hacer N multiplicaciones

Algoritmo factorial:
Leer N
factorial ← 1
factor ← 1
repetir N veces:
  factorial ← factorial * factor
  factor ← factor + 1
Mostrar factorial en pantalla

PROGRAMA PARA "FACTORIAL"

```
PROGRAM CalcularFactorial;
{calcula factorial de un número ingresado}
VAR
numero, factor, factorial: INTEGER;
BEGIN
writeln('ingrese número >= 0');
readln(numero);
factorial:=1;
FOR factor:=1 TO numero DO
  factorial:=factorial * factor;
Writeln(' El factorial de ',numero,
' es ' , factorial);
END.
```

numero	factor	factorial
?	?	?
5	1	1
	2	2
	3	6
	4	24
	5	120

PROBLEMAS PARA PRACTICAR

Problema: Escriba un programa que dados tres valores enteros V1, V2 y N ingresados por el usuario, muestre y cuente cuantos enteros hay entre V1 y V2 que sean múltiplos de N.

```
Ingrese dos valores enteros: 7 30
Ingrese un divisor: 6
Entre 7 y 30, son múltiplos de 6:
12, 18, 24, 30
En total son 4 múltiplos.
```


OBSERVACIONES SOBRE REPETICIÓN INCONDICIONAL

```
FOR V:= exp1 DOWNTO exp2 DO
  <sentencia>
```

- Tanto en un ciclo FOR-TO como en FOR-DOWNTO, el valor de *Exp1* (llamado *valor inicial*) y el de *Exp2* (llamado *valor final*) deben poder calcularse al comenzar la repetición, de lo contrario es un **error de programación.**

```
PROGRAM Incorrecto;
{¿Por qué es incorrecto?}
VAR
v, inicio, tope: integer;
BEGIN
tope:=10;
FOR v := inicio TO tope DO
  writeln(v);
END.
```

MAL

V	inicio	tope
?	?	?
		10

Error de programación: inicio sin valor

MUY IMPORTANTE

```
FOR V:= <exp1> DOWNTO <exp2> DO
  <Bloque de sentencias>
```

- Ya sea para un ciclo **FOR-TO** o **FOR-DOWNTO**, en RPA será considerado error de programación:
  1) Cambiar el valor de la variable de control V, dentro del bloque de sentencias de un ciclo FOR.
  2) Cambiar el valor de cualquier variable de <expresión1> o <expresión2>, dentro del bloque de sentencias de un FOR, con motivo de que cambien los límites de la iteración.

- Si surge la necesidad de hacerlo es porque tendría usar una repetición condicional con **REPEAT** o **WHILE** (que veremos muy pronto…).

ERROR DE PROGRAMACIÓN

```
…
FOR  V:= 1 TO 100 DO
begin
   writeln(V);
   V:= V + 5;
end;
…
```

MAL

```
…
FOR  V:= 1 TO 100 DO
begin
   writeln(V);
   if V=12 then V:=100;
end;
…
```

MAL

Error de programación

Importante: es un error de programación intentar controlar "manualmente" la variable de control o los límites de un for.

Lazarus dice: *Error: Illegal assignment to for-loop variable "v"*
Traducido: Error: asignación ilegal a la variable de control "v"

ERROR DE PROGRAMACIÓN

```
…
ultimo:=100;
FOR  V:= 1 TO ultimo DO
begin
   writeln(V);
   if V=12 then
      ultimo:=13;
end;
…
```

MAL

Es un error pensar que cambiando el valor del límite, la cantidad de repeticiones de un FOR puede cambiar o dejar de repetir.

Importante: es un error de programación intentar controlar "manualmente" la variable de control o los límites de un FOR.

REPETICIONES ANIDADAS

```
FOR v:= 1 TO 3 DO
  writeln(v);
```

¿cuántas veces se ejecuta **writeln(v)**?

```
FOR v:= 1 TO 3 DO
  FOR h:= 1 TO 2 DO
    writeln(v, h);
```

¿cuántas veces se ejecuta **writeln(v, h)**?

```
FOR v:= 1 TO 3 DO
  FOR h:= 1 TO 2 DO
    FOR t:= 1 DOWNTO 2 DO
      writeln(v, h, t);
```

Obs: usan diferentes variables de control ¿Por qué?

¿cuántas veces se ejecuta **writeln(v,h,t)**?

REPETICIONES ANIDADAS: EJEMPLO

```
PROGRAM combinaLetras;
{muestra las 27 combinaciones
de las letras A B y C}
VAR L1,L2,L3: CHAR;
BEGIN
FOR L1 := 'A' TO 'C' DO
   FOR L2 := 'A' TO 'C' DO
      FOR L3 := 'A' TO 'C' DO
         writeln (L1,L2,L3)
END.
```

```
AAA
AAB
AAC
ABA
ABB
ABC
...
```

L1	L2	L3
?	?	?
A	A	A
A	A	B
A	A	C
A	B	A

- **Problema propuesto**: Un dominio automotor (patente) es una combinación de letras y dígitos. Escriba un programa que muestre TODAS las patentes posibles. ¿Cuántas son?

SENTENCIAS FOR-TO ANIDADAS. EJEMPLO.

```
PROGRAM FOR_anidados;
{ejemplo de bucles FOR anidados donde el ciclo
interno tiene un límite (cant) que va
cambiando en cada iteración}
VAR
Letra: char; i,cant: integer;
BEGIN
cant:=1; {cantidad de letras por renglón}
FOR Letra := 'A' TO 'E' DO
  BEGIN  {imprime un renglón de "Letra"s}
    FOR i := 1 TO cant DO
      write(Letra);
    writeln; {baja de renglón}
    cant:=cant+1;{incr. la cant por renglón}
  END;
END.
```

```
A
BB
CCC
DDDD
EEEEE
```

Realice la TRAZA

```
PROGRAM FOR_anidados2;
VAR cant, num, pri, ult : integer;
BEGIN {otro ejemplo de límites que cambian}
writeln('ingrese 2 dig.');
readln(pri, ult);
{primera mitad incrementando}
FOR cant := pri  TO ult DO
  BEGIN
    FOR num := pri TO cant DO
      write(num);
    writeln;
  END;
{segunda mitad decrementando}
FOR cant := ult-1 DOWNTO pri DO
  BEGIN
    FOR num := pri TO cant DO
      write(num);
    writeln;
  END;
END.
```

```
ingrese 2 dig.:
 2 5

 2
 23
 234
 2345
 234
 23
 2
```

PROBLEMAS PARA PRACTICAR

- **Problema propuesto**: Escriba un programa que pida al usuario un valor 0 < N < 10 y dibuje una forma como la que sigue (por ejemplo para N=4):

```
ingrese N: 4

1
22
333
4444
333
22
1
```

1. Entender el problema
2. Buscar solución:
   3. Buscar ejemplos particulares
   4. Dividir el problema en partes
5. Escribir algoritmo
6. Verificar con una traza.
7. Escribir programa
8. Verificar con una traza.

38

USOS PERMITIDOS DE LA VARIABLE DE CONTROL

- La variable de control si puede usarse y modificarse en otras partes del programa que estén “afuera” del ciclo FOR.
- Como muestra el ejemplo 1, si es posible cambiar la variable de control V antes o después del ciclo FOR.
- En el ejemplo 2 se muestra que la misma variable de control V también usarse como variable de control para otro ciclo FOR siempre que uno no esté anidado en el otro.

```
{ej. 1: válido}
V:= 9;
writeln(V);
FOR V:= 1 TO 3 DO
  writeln(V);
V:= 8;
writeln(V);
```

```
{ej 2: válido}
FOR V:= 1 TO 3 DO
  writeln(V);
FOR V:= 4 TO 9 DO
  writeln(V);
V:= 4;
writeln(V);
```

```
{ej 3: inválido}
FOR V:= 1 TO 3 DO
begin
  FOR  V:= 4 TO 9 DO
    writeln(V);
end;
```

MAL